

**"Systems and Methods for Generating Visual
Representations of Graphical Data and Digital
Document Processing"**

Field of the Invention

The invention relates to data processing methods and systems. More particularly, the invention relates to methods and systems for processing "graphical data" and "digital documents" (as defined herein) and to devices incorporating such methods and systems. In general terms, the invention is concerned with generating output representations of source data and documents; e.g. as a visual display or as hardcopy.

Background to the Invention

As used herein, the terms "graphical data", "graphical object" and "digital document" are used to describe a digital representation of any type of data processed by a data processing system which is intended, ultimately, to be output in some form, in whole or in part, to a human user, typically by being displayed or reproduced visually (e.g. by means of a visual display unit or printer), or by text-to-speech conversion, etc. Such data, objects and documents may include any features capable of representation, including but not limited to the following: text; graphical images; animated graphical images; full motion video images; interactive icons, buttons, menus or hyperlinks. A

1 digital document may also include non-visual
2 elements such as audio (sound) elements. A digital
3 document generally includes or consists of graphical
4 data and/or at least one graphical object.

5
6 Data processing systems, such as personal
7 computer systems, are typically required to process
8 "digital documents", which may originate from any
9 one of a number of local or remote sources and which
10 may exist in any one of a wide variety of data
11 formats ("file formats"). In order to generate an
12 output version of the document, whether as a visual
13 display or printed copy, for example, it is
14 necessary for the computer system to interpret the
15 original data file and to generate an output
16 compatible with the relevant output device (e.g.
17 monitor, or other visual display device, or
18 printer). In general, this process will involve an
19 application program adapted to interpret the data
20 file, the operating system of the computer, a
21 software "driver" specific to the desired output
22 device and, in some cases (particularly for monitors
23 or other visual display units), additional hardware
24 in the form of an expansion card.

25
26 This conventional approach to the processing of
27 digital documents in order to generate an output is
28 inefficient in terms of hardware resources, software
29 overheads and processing time, and is completely
30 unsuitable for low power, portable data processing
31 systems, including wireless telecommunication
32 systems, or for low cost data processing systems

09835483-04-1601

such as network terminals, etc. Other problems are encountered in conventional digital document processing systems, including the need to configure multiple system components (including both hardware and software components) to interact in the desired manner, and inconsistencies in the processing of identical source material by different systems (e.g. differences in formatting, colour reproduction, etc). In addition, the conventional approach to digital document processing is unable to exploit the commonality and/or re-usability of file format components.

Summary of the Invention

It is an object of the present invention to provide methods and systems for processing graphical data, graphical objects and digital documents, and devices incorporating such methods and systems, which obviate or mitigate the aforesaid disadvantages of conventional methods and systems.

The invention, in its various aspects, is defined in the Claims appended hereto. Further aspects and features of the invention will be apparent from the following description.

In a first aspect, the invention relates to a method of redrawing a visual display of graphical data whereby a current display is replaced by an updated display, comprising, in response to a redraw request, immediately replacing the current display

1 with a first approximate representation of the
2 updated display, generating a final updated display,
3 and replacing the approximate representation with
4 the final updated display.

5
6 In a second aspect, the invention relates to a
7 method of generating variable visual representations
8 of graphical data, comprising dividing said
9 graphical data into a plurality of bitmap tiles of
10 fixed, predetermined size, storing said tiles in an
11 indexed array and assembling a required visual
12 representation of said graphical data from a
13 selected set of said tiles.

14
15 The methods of said second aspect may be
16 employed in methods of the first aspect.

17
18 A third aspect of the invention relates to a
19 method of processing a digital document, said
20 document comprising a plurality of graphical objects
21 arranged on at least one page, comprising dividing
22 said document into a plurality of zones and, for
23 each zone, generating a list of objects contained
24 within and overlapping said zone.

25
26 The methods of the second aspect may be
27 employed in the methods of the third aspect.

28
29 In accordance with a fourth aspect of the
30 invention, there is provided a digital document
31 processing system adapted to implement the methods
32 of any of the first to third aspects.

0965463, 041601

1
2 A preferred system in accordance with the
3 fourth aspect of the invention comprises:

4 an input mechanism for receiving an input
5 bytestream representing source data in one of a
6 plurality of predetermined data formats;

7 an interpreting mechanism for interpreting said
8 bytestream;

9 a converting mechanism for converting
10 interpreted content from said bytestream into an
11 internal representation data format; and

12 a processing mechanism for processing said
13 internal representation data so as to generate
14 output representation data adapted to drive an
15 output device.

16
17 In a further aspect, the invention relates to a
18 graphical user interface for a data processing
19 system in which interactive visual displays employed
20 by the user interface are generated by means of a
21 digital document processing system in accordance
22 with the fourth aspect of the invention and to data
23 processing systems incorporating such a graphical
24 user interface.

25
26 In still further aspects, the invention relates
27 to various types of device incorporating a digital
28 document processing system in accordance with the
29 fourth aspect of the invention, including hardware
30 devices, data processing systems and peripheral
31 devices.

32

09835483-041601

1 Embodiments of the invention will now be
2 described, by way of example only, with reference to
3 the accompanying drawings.

4
5 **Brief Description of the Drawings**

6
7 Fig. 1 is a block diagram illustrating an
8 embodiment of a preferred digital document
9 processing system which may be employed in
10 implementing various aspects of the present
11 invention;

12
13 Fig. 2A is a flow diagram illustrating a first
14 embodiment of a first aspect of the present
15 invention;

16
17 Fig. 2B is a flow diagram illustrating a second
18 embodiment of a first aspect of the present
19 invention;

20
21 Fig. 3 is a diagram illustrating a method of
22 scaling a bitmap in a preferred embodiment of the
23 first aspect of the invention;

24
25 Fig. 4A is a diagram illustrating a
26 conventional method of using an off-screen buffer
27 for panning a visual display of a digital document;

28
29 Fig. 4B is a diagram illustrating a method of
30 using an off-screen buffer for panning a visual
31 display of a digital document in accordance with a
32 second aspect of the present invention;

0933483-041601

Fig. 5A is a diagram illustrating memory allocation and fragmentation associated with the conventional method of Fig. 4A;

Fig. 5B is a diagram illustrating memory allocation and fragmentation associated with the method of Fig. 4B;

Fig. 5C is a diagram illustrating a preferred method of implementing the method of Fig. 4B.

Fig. 6 is a diagram illustrating the use of multiple parallel processor modules for implementing the method of Fig. 4B; and

Figs. 7 and 8 are diagrams illustrating a method of processing a digital document in accordance with a third aspect of the invention.

Detailed Description of the Preferred Embodiments

Referring now to the drawings, Fig. 1 illustrates a preferred digital document processing system 8 in which the methods of the various aspects of the present invention may be implemented. Before describing the methods of the invention in detail, the system 8 will first be described by way of background. It will be understood that the methods of the present invention may be implemented in processing systems other than the system 8 as described herein.

1
2 In general terms, the system 8 will process one
3 or more source documents 10 comprising data files in
4 known formats. The input to the system 8 is a
5 bytestream comprising the content of the source
6 document. An input module 11 identifies the file
7 format of the source document on the basis of any
8 one of a variety of criteria, such as an explicit
9 file-type identification within the document, from
10 the file name (particularly the file name
11 extension), or from known characteristics of the
12 content of particular file types. The bytestream is
13 input to a "document agent" 12, specific to the file
14 format of the source document. The document agent 12
15 is adapted to interpret the incoming bytestream and
16 to convert it into a standard format employed by the
17 system 8, resulting in an internal representation 14
18 of the source data in a "native" format suitable for
19 processing by the system 8. The system 8 will
20 generally include a plurality of different document
21 agents 12, each adapted to process one of a
22 corresponding plurality of predetermined file
23 formats.

24
25 The system 8 may also be applied to input
26 received from an input device such as a digital
27 camera or scanner. In this case the input
28 bytestream may originate directly from the input
29 device, rather than from a "source document" as
30 such. However, the input bytestream will still be
31 in a predictable data format suitable for processing
32 by the system and, for the purposes of the system,

1 input received from such an input device may be
2 regarded as a "source document".

3
4 The document agent 12 employs a library 16 of
5 standard objects to generate the internal
6 representation 14, which describes the content of
7 the source document in terms of a collection of
8 generic objects whose types are as defined in the
9 library 16, together with parameters defining the
10 properties of specific instances of the various
11 generic objects within the document. It will be
12 understood that the internal representation may be
13 saved/stored in a file format native to the system
14 and that the range of possible source documents 10
15 input to the system 8 may include documents in the
16 system's native file format. It is also possible
17 for the internal representation 14 to be converted
18 into any of a range of other file formats if
19 required, using suitable conversion agents (not
20 shown).

21
22 The generic objects employed in the internal
23 representation 14 will typically include: text,
24 bitmap graphics and vector graphics (which may or
25 may not be animated and which may be two- or three-
26 dimensional), video, audio, and a variety of types
27 of interactive object such as buttons and icons.
28 The parameters defining specific instances of
29 generic objects will generally include dimensional
30 co-ordinates defining the physical shape, size and
31 location of the object and any relevant temporal
32 data for defining objects whose properties vary with

09035483 04-1604

1 time (allowing the system to deal with dynamic
2 document structures and/or display functions). For
3 text objects, the parameters will normally also
4 include a font and size to be applied to a character
5 string. Object parameters may also define other
6 properties, such as transparency.

7
8 The format of the internal representation 14
9 separates the "structure" (or "layout") of the
10 documents, as described by the object types and
11 their parameters, from the "content" of the various
12 objects; e.g. the character string (content) of a
13 text object is separated from the dimensional
14 parameters of the object; the image data (content)
15 of a graphic object is separated from its
16 dimensional parameters. This allows document
17 structures to be defined in a very compact manner
18 and provides the option for content data to be
19 stored remotely and to be fetched by the system only
20 when needed.

21
22 The internal representation 14 describes the
23 document and its constituent objects in terms of
24 "high-level" descriptions.

25
26 The internal representation data 14 is input to
27 a parsing and rendering module 18 which generates a
28 context-specific representation 20 or "view" of the
29 document represented by the internal representation
30 14. The required view may be of the whole document
31 or of part(s) (subset(s)) thereof. The
32 parser/renderer 18 receives view control inputs 40

1 which define the viewing context and any related
2 temporal parameters of the specific document view
3 which is to be generated. For example, the system
4 may be required to generate a zoomed view of part of
5 a document, and then to pan or scroll the zoomed
6 view to display adjacent portions of the document.
7 The view control inputs 40 are interpreted by the
8 parser/renderer 18 in order to determine which parts
9 of the internal representation are required for a
10 particular view and how, when and for how long the
11 view is to be displayed.

12

13 The context-specific representation/view 20 is
14 expressed in terms of primitive shapes and
15 parameters.

16

17 The parser/renderer 18 may also perform
18 additional pre-processing functions on the relevant
19 parts of the internal representation 14 when
20 generating the required view 20 of the source
21 document 10. The view representation 20 is input to
22 a shape processor module 22 for final processing to
23 generate a final output 24, in a format suitable for
24 driving an output device 26 (or multiple output
25 devices), such as a display device or printer.

26

27 The pre-processing functions of the
28 parser/renderer 18 may include colour correction,
29 resolution adjustment/enhancement and anti-aliasing.
30 Resolution enhancement may comprise scaling
31 functions which preserve the legibility of the
32 content of objects when displayed or reproduced by

1 the target output device. Resolution adjustment may
2 be context-sensitive; e.g. the display resolution of
3 particular objects may be reduced while the
4 displayed document view is being panned or scrolled
5 and increased when the document view is static (as
6 described further below in relation to the first
7 aspect of the invention).

8
9 There may be a feedback path 42 between the
10 renderer/parser 18 and the internal representation
11 14; e.g. for the purpose of triggering an update of
12 the content of the internal representation 14, such
13 as in the case where the document 10 represented by
14 the internal representation comprises a multi-frame
15 animation.

16
17 The output representation 20 from the
18 parser/renderer 18 expresses the document in terms
19 of "primitive" objects. For each document object,
20 the representation 20 preferably defines the object
21 at least in terms of a physical, rectangular
22 boundary box, the actual shape of the object bounded
23 by the boundary box, the data content of the object,
24 and its transparency.

25
26 The shape processor 22 interprets the
27 representation 20 and converts it into an output
28 frame format 24 appropriate to the target output
29 device 26; e.g. a dot-map for a printer, vector
30 instruction set for a plotter, or bitmap for a
31 display device. An output control input 44 to the
32 shape processor 22 defines the necessary parameters

1 for the shape processor 22 to generate output 24
2 suitable for a particular output device 26.

3
4 The shape processor 22 preferably processes the
5 objects defined by the view representation 20 in
6 terms of "shape" (i.e. the outline shape of the
7 object), "fill" (the data content of the object) and
8 "alpha" (the transparency of the object), performs
9 scaling and clipping appropriate to the required
10 view and output device, and expresses the object in
11 terms appropriate to the output device (typically in
12 terms of pixels by scan conversion or the like, for
13 most types of display device or printer).

14
15 The shape processor 22 preferably includes an
16 edge buffer which defines the shape of an object in
17 terms of scan-converted pixels, and preferably
18 applies anti-aliasing to the outline shape. Anti-
19 aliasing is preferably performed in a manner
20 determined by the characteristics of the output
21 device 26 (i.e. on the basis of the control input
22 44), by applying a grey-scale ramp across the object
23 boundary. This approach enables memory efficient
24 shape-clipping and shape-intersection processes.

25
26 A look-up table may be employed to define
27 multiple tone response curves, allowing non-linear
28 rendering control (gamma correction).

29
30 The individual objects processed by the shape
31 processor 22 are combined in the composite output
32 frame 24. The quality of the final output can also

1 be controlled by the user via the output control
2 input 44.

3
4 The shape processor 22 has a multi-stage
5 pipeline architecture which lends itself to parallel
6 processing of multiple objects, or of multiple
7 documents, or of multiple subsets of one or more
8 document, by using multiple instances of the shape
9 processor pipeline. The pipeline architecture is
10 also easily modified to include additional
11 processing functions (e.g. filter functions) if
12 required. Outputs from multiple shape processors 22
13 may generate multiple output frames 24 or may be
14 combined in a single output frame 24.

15
16 The system architecture is modular in nature.
17 This enables, for example, further document agents
18 to be added as and when required, to deal with
19 additional source file formats. The modular
20 architecture also allows individual modules such as
21 the library 16, parser/renderer 18 or shape
22 processor 22 to be modified or upgraded without
23 requiring changes to other modules.

24
25 The system architecture as a whole also lends
26 itself to parallelism in whole or in part for
27 simultaneous processing of multiple input documents
28 10a, 10b etc. or subsets of documents, in one or
29 more file formats, via one or more document agents
30 12, 12a. The integrated, modular nature of the
31 system allows multiple instances of system modules
32 to be spawned within a data processing system or

09835433-041601

1 multiple levels of display quality for this purpose.
2 Typically, this will involve pre-built, low
3 resolution bitmap representations of document
4 objects and/or dynamically built and scaled bitmaps,
5 with or without interpolation. This approach
6 provides a highly responsive display which makes
7 best use of available memory/bandwidth.

8
9 Methods embodying this first aspect of the
10 present invention are illustrated in Figs. 2A and
11 2B.

12
13 When a redraw request is initiated within the
14 system, it is necessary for all or part of the
15 current frame to be re-rendered and displayed. The
16 process of re-rendering the frame may take a
17 significant amount of time.

18
19 Referring to Fig. 2A, when a redraw request 100
20 is initiated, the output frame is immediately
21 updated (102) using one or more reduced resolution
22 ("thumbnail") bitmap representations of the document
23 or parts thereof which are scaled to approximate the
24 required content of the redrawn display. In the
25 system 8 of Fig. 1, the bitmap representation(s)
26 employed for this purpose may be pre-built by the
27 parser/renderer 18 and stored for use in response to
28 redraw requests. This approximation of the redrawn
29 display can be generated much more quickly than the
30 full re-rendering of the display, providing a
31 temporary display while re-rendering is completed.
32 In the embodiment of Fig. 2A, the full re-rendering

09835453-041604

1 of the display (104) is performed in parallel with
2 the approximate redraw 102, and replaces the
3 temporary display once it is complete (106). The
4 method may include one or more additional
5 intermediate updates 108 of the approximate
6 temporary display while the full re-rendering 104 is
7 completed. These intermediate updates may
8 progressively "beautify" the temporary display (i.e.
9 provide successively better approximations of the
10 final display); e.g. by using better quality scaled
11 bitmaps and/or by superimposing vector outlines of
12 objects on the bitmap(s).

13
14 The method of Fig. 2A also allows the redraw
15 process to be interrupted by a new redraw request
16 (110). The full re-rendering process 104 can simply
17 be halted and the system processes the new redraw
18 request as before.

19
20 Fig. 2B illustrates an alternative embodiment
21 in which a redraw request 112 is followed by an
22 approximate thumbnail-based redraw 114 as before,
23 and the full frame redraw 116 follows in series
24 after the approximate redraw (instead of in parallel
25 as in Fig. 2A) to generate the final full resolution
26 display 118. This process may also be interrupted
27 at any stage by a new redraw request.

28
29 The methods of Figs. 2A and 2B may be applied
30 to all types of redraw requests, including screen
31 re-builds, scrolling, panning and scaling (zooming).

32

Fig. 3 illustrates a preferred method of zooming/scaling a thumbnail bitmap. A basic bitmap 120 is created and stored by the system at some previous stage of document processing as previously described. Assuming that the bitmap is required to be scaled by some arbitrary factor (e.g. by a factor 4.4), the basic thumbnail 120 is scaled in two stages: first, the thumbnail is scaled by a fractional amount (122) corresponding to the final scaling factor divided by the whole number part thereof (4.4 divided by 4 equals 1.1 in this example), and then by an integer amount (124) corresponding to the whole number part of the final scaling factor (i.e. x4 in this example). This is faster than a single stage zoom of 4.4, at the expense of a small increase in memory requirement.

The scaling operations described above may be performed with or without interpolation. Fig. 3 shows the final zoomed bitmap 124 interpolated to provide a resolution of 16 x 16 as compared with the original 8 x 8 bitmap 120. Interpolation may be performed using any of a variety of well known interpolation methods.

The ability to process transparent objects is a significant feature of the system of Fig. 1. However, this necessitates the use of off-screen buffering in the shape processor 22 in order to assemble a final output frame. Typically, as shown in Fig. 4A, a conventional off-screen buffer 130 will cover an area larger than the immediate display

1 area, allowing a limited degree of panning/scrolling
2 within the buffer area, but the entire buffer has to
3 be re-centred and re-built when the required display
4 moves outwith these limits. This requires a block-
5 copy operation within the buffer and redrawing of
6 the remaining "dirty rectangle" (132) before block-
7 copying the updated buffer contents to the screen
8 134.

9
10 In accordance with a second aspect of the
11 present invention, as illustrated in Fig. 4B, the
12 efficiency of such buffering processes is improved
13 by defining the buffer content as an array of tiles
14 136, indexed in an ordered list. Each tile
15 comprises a bitmap of fixed size (e.g. 32 x 32 or 64
16 x 64) and may be regarded as a "mini-buffer". When
17 the required display view moves outwith the buffer
18 area, it is then only necessary to discard those
19 tiles which are no longer required, build new tiles
20 to cover the new area of the display and update the
21 tile list (138). This is faster and more efficient
22 than conventional buffering processes, since no
23 block-copying is required within the buffer and no
24 physical memory is required to be moved or copied.

25
26 The tiling scheme described may be used
27 globally to provide a tilepool for all document and
28 screen redraw operations. The tiles are used to
29 cache the document(s) off-screen and allow rapid,
30 efficient panning and re-centering of views.

31

1 The use of a tilepool as described also allows
2 for more efficient usage of memory and processor
3 resources. Fig. 5A shows how conventional off-
4 screen buffering methods, involving data blocks
5 which have arbitrary, unpredictable sizes, result in
6 the fragmentation of memory due to unpredictable
7 contiguous block allocation requirements. Blocks of
8 memory required by buffering operations are mis-
9 matched with processor memory management unit (MMU)
10 blocks, so that re-allocation of memory becomes
11 inefficient, requiring large numbers of physical
12 memory copy operations, and cache consistency is
13 impaired. By using tiles of fixed, predetermined
14 size, memory requirements become much more
15 predictable, so that memory and processor resources
16 may be used and managed much more efficiently,
17 fragmentation may be unlimited without affecting
18 usability and the need for memory copy operations
19 may be substantially eliminated for many types of
20 buffering operations. Ideally, the tile size is
21 selected to correspond with the processor MMU block
22 size.

23
24 Fig. 5C illustrates a preferred scheme for
25 managing tiles within the tilepool. Tile number
26 zero is always reserved for building each new tile.
27 Once a new tile has been built, it is re-numbered
28 using the next available free number (i.e. where the
29 tilepool can accommodate a maximum of n tiles the
30 number of tile addresses allocated is restricted to
31 $n-1$). In the event of a tilepool renumbering
32 failure, when the tilepool is full and there are no

1 more free tiles, the new tile 0 is written directly
2 to screen and a background process (thread) is
3 initiated for garbage collection (e.g.
4 identification and removal of "aged" tiles) and/or
5 allocation of extra tile addresses. This provides
6 an adaptive mechanism for dealing with resource-
7 allocation failures.

8
9 The tiling scheme described lends itself to
10 parallel processing, as illustrated in Fig. 6.
11 Processing of a set of tiles can be divided between
12 multiple parallel processes (e.g. between multiple
13 instances of the shape processor 22 of Fig. 1). For
14 example, a set of tiles 1-20 can be divided based on
15 the screen positions of the tiles so that the
16 processing of tiles 1-10 is handled by one processor
17 WASP-A and the processing of tiles 11-20 is handled
18 by a second processor WASP-B. Accordingly, if a
19 redraw instruction requires tiles 1-3, 7-9 and 12-14
20 to be redrawn, tiles 2-4 and 7-9 are handled by
21 WASP-A and tiles 12-14 by WASP-B. Alternatively,
22 the set of tiles can be divided based on the
23 location of the tiles in a tile memory map, dividing
24 the tile memory into memory A for processing by
25 WASP-A and memory B for processing by WASP-B.

26
27 The tiling scheme described facilitates the use
28 of multiple buffering and off-screen caching. It
29 also facilitates interruptable re-draw functions
30 (e.g. so that a current re-draw may be interrupted
31 and a new re-draw initiated in response to user
32 input), efficient colour/display conversion and

1 dithering, fast 90 degree (portrait/landscape)
2 rotation of whole display in software, and reduces
3 the redraw memory required for individual objects.
4 Tiling also makes interpolated bitmap scaling faster
5 and more efficient. It will also be appreciated
6 that a system such as that of Fig. 1 may employ a
7 common tilepool for all operating system/GUI and
8 application display functions of a data processing
9 system.

10
11 It will be understood that the tiling methods
12 of the second aspect of the invention may
13 advantageously be combined with the redraw methods
14 of the first aspect of the invention.

15
16 In accordance with a third aspect of the
17 present invention, the processing of a document
18 involves dividing each page of the document to be
19 viewed into zones (this would involve interaction of
20 the renderer/parser 18 and shape processor 22 in the
21 system 8 of Fig. 1), as illustrated in Fig. 7. Each
22 zone A, B, C, D has associated with it a list of all
23 objects 1-8 contained within or overlapping that
24 zone. Re-draws can then be processed on the basis
25 of the zones, so that the system need only process
26 objects associated with the relevant zones affected
27 by the re-draw. This approach facilitates parallel
28 processing and improves efficiency and reduces
29 redundancy. The use of zones also facilitates the
30 use of the system to generate different outputs for
31 different display devices (e.g. for generating a

0935483-041601

1 composite/mosaic output for display by an array of
2 separate display screens).

3
4 As illustrated in Fig. 8, without the use of
5 zoning, any screen update relating to the shaded
6 area 142 would require each of the eight objects 1
7 to 8 to be checked to see whether the bounding box
8 of the object intersects the area 142 in order to
9 determine whether that object needs to be plotted.
10 With zoning, it is possible to determine firstly
11 which zones intersect the area 142 (zone D only in
12 this example), secondly, which objects intersect the
13 relevant zone(s) (object 2 only in this case), and
14 then it is only necessary to check whether the
15 bounding boxes of those objects which intersect the
16 relevant zone(s) also intersect the area 142. In
17 many cases, this will greatly reduce the overhead
18 involved in extracting and comparing objects with
19 the area 142 of interest.

20
21 It will be appreciated that zoning of this type
22 may be of little or no benefit in some circumstances
23 (e.g. in the extreme case where all objects on a
24 page intersect all zones of the page); the
25 relationship between the zone size and the typical
26 object size may be significant in this respect. For
27 the purposes of determining the nature of any zoning
28 applied to a particular page, an algorithm may be
29 employed to analyse the page content and to
30 determine a zoning scheme (the number, size and
31 shape of zones) which might usefully be employed for
32 that page. However, for typical page content, which

1 will commonly include many small, locally clustered
2 objects, an arbitrary division of the page into
3 zones is likely to yield significant benefits.

4
5 The zoning and tiling schemes described above
6 are independent in principle but may be combined
7 advantageously; i.e. zones may correlate with one or
8 more tiles. Again this facilitates parallelism and
9 optimises use of system resources.

10

11 Referring again to Fig. 1, the system
12 preferably employs a device-independent colour
13 model, suitably a luminance/chrominance model such
14 as the CIE L*a*b* 1976 model. This reduces
15 redundancy in graphic objects, improves data
16 compressibility and improves consistency of colour
17 output between different output devices. Device-
18 dependent colour correction can be applied on the
19 basis of the device-dependent control input 44 to
20 the shape processor 22.

21

22 Fig. 1 shows the system having an input end at
23 which the source bytestream is received and an
24 output end where the final output frame 24 is output
25 from the system. However, it will be understood
26 that the system may include intermediate inputs and
27 outputs at other intermediate stages, such as for
28 fetching data content or for saving/converting data
29 generated in the course of the process.

30

31 Digital document processing systems in
32 accordance with the fourth aspect of the present

1 invention may be incorporated into a variety of
2 types of data processing systems and devices, and
3 into peripheral devices, in a number of different
4 ways.

5
6 In a general purpose data processing system
7 (the "host system"), the system of the present
8 invention may be incorporated alongside the
9 operating system and applications of the host system
10 or may be incorporated fully or partially into the
11 host operating system.

12
13 For example, the system of the present
14 invention enables rapid display of a variety of
15 types of data files on portable data processing
16 devices with LCD displays without requiring the use
17 of browsers or application programs. This class of
18 data processing devices requires small size, low
19 power processors for portability. Typically, this
20 requires the use of advanced RISC-type core
21 processors designed into ASICs (application specific
22 integrated circuits), in order that the electronics
23 package is as small and highly integrated as
24 possible. This type of device also has limited
25 random access memory and typically has no non-
26 volatile data store (e.g. hard disk). Conventional
27 operating system models, such as are employed in
28 standard desktop computing systems (PCs), require
29 high powered central processors and large amounts of
30 memory in order to process digital documents and
31 generate useful output, and are entirely unsuited
32 for this type of data processing device. In

09825483.041604

1 primary user interface comprises a graphical display
2 with which the user interacts directly by means of a
3 stylus device), internet-enabled mobile telephones
4 and other communications devices, etc.

5
6 The system may also be incorporated into low
7 cost data processing terminals such as enhanced
8 telephones and "thin" network client terminals (e.g.
9 network terminals with limited local processing and
10 storage resources), and "set-top boxes" for use in
11 interactive/internet-enabled cable TV systems.

12
13 When integrated with the operating system of a
14 data processing system, the system of the present
15 invention may also form the basis of a novel
16 graphical user interface (GUI) for the operating
17 system (OS). Documents processed and displayed by
18 the system may include interactive features such as
19 menus, buttons, icons etc. which provide the user
20 interface to the underlying functions of the
21 operating system. By extension, a complete OS/GUI
22 may be expressed, processed and displayed in terms
23 of system "documents". The OS/GUI could comprise a
24 single document with multiple "chapters".

25
26 The system of the present invention may also be
27 incorporated into peripheral devices such as
28 hardcopy devices (printers and plotters), display
29 devices (such as digital projectors), networking
30 devices, input devices (cameras, scanners etc.) and
31 also multi-function peripherals (MFPs).

32

09835463-04-1601

1 When incorporated into a printer, the system
2 may enable the printer to receive raw data files
3 from the host data processing system and to
4 reproduce the content of the original data file
5 correctly, without the need for particular
6 applications or drivers provided by the host system.
7 This avoids the need to configure a computer system
8 to drive a particular type of printer. The present
9 system may directly generate a dot-mapped image of
10 the source document suitable for output by the
11 printer (this is true whether the system is
12 incorporated into the printer itself or into the
13 host system). Similar considerations apply to other
14 hardcopy devices such as plotters.

15
16 When incorporated into a display device, such
17 as a projector, the system may again enable the
18 device to display the content of the original data
19 file correctly without the use of applications or
20 drivers on the host system, and without the need for
21 specific configuration of the host system and/or
22 display device. Peripheral devices of these types,
23 when equipped with the present system, may receive
24 and output data files from any source, via any type
25 of data communications network.

26
27 From the foregoing, it will be understood that
28 the system of the present invention may be "hard-
29 wired; e.g. implemented in ROM and/or integrated
30 into ASICs or other single-chip systems, or may be
31 implemented as firmware (programmable ROM such as
32 flashable ePROM), or as software, being stored

1 locally or remotely and being fetched and executed
2 as required by a particular device.

3

4 Improvements and modifications may be
5 incorporated without departing from the scope of the
6 present invention.

09835483-041604